PSTAT-131-HW3

Ephets Head

4/15/2022

For this assignment, we will be using the *titanic* data set to predict which passengers would survive the Titanic shipwreck. First we will load the data into R from *data/titanic.csv*.

```
titanic <-read_csv("data/titanic.csv", show_col_types=FALSE)
titanic
```

```
## # A tibble: 891 x 12
##    passenger_id survived pclass name       sex     age sib_sp parch ticket  fare
##           <dbl> <chr>     <dbl> <chr>      <chr> <dbl>  <dbl> <dbl> <chr>  <dbl>
##  1            1 No            3 Braund, M~ male     22      1     0 A/5 2~  7.25
##  2            2 Yes           1 Cumings, ~ fema~    38      1     0 PC 17~ 71.3
##  3            3 Yes           3 Heikkinen~ fema~    26      0     0 STON/~  7.92
##  4            4 Yes           1 Futrelle,~ fema~    35      1     0 113803 53.1
##  5            5 No            3 Allen, Mr~ male     35      0     0 373450  8.05
##  6            6 No            3 Moran, Mr~ male     NA      0     0 330877  8.46
##  7            7 No            1 McCarthy,~ male     54      0     0 17463  51.9
##  8            8 No            3 Palsson, ~ male      2      3     1 349909 21.1
##  9            9 Yes           3 Johnson, ~ fema~    27      0     2 347742 11.1
## 10           10 Yes           2 Nasser, M~ fema~    14      1     0 237736 30.1
## # ... with 881 more rows, and 2 more variables: cabin <chr>, embarked <chr>
```

The variables *survived* and *pclass* are both categorical predictors not representative of a quantity, so they must be changed into factor variables.

```
titanic$survived <- factor(titanic$survived)

#re-assign the base level of the factored variable "survived" to be "Yes"
titanic$survived <- relevel(titanic$survived, "Yes")

titanic$pclass <- factor(titanic$pclass)
```

Question 1: Split the data, stratifying the outcome variable, *survived*. Verify that the training and data sets have appropriate numbers of observations. Note any issues with the training set, such as missing data. Why is it a good idea to use stratified sampling for this data?

```
set.seed(2022)

titanic_split <- initial_split(titanic, prop=0.7, strata=survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

dim(titanic_split)
```

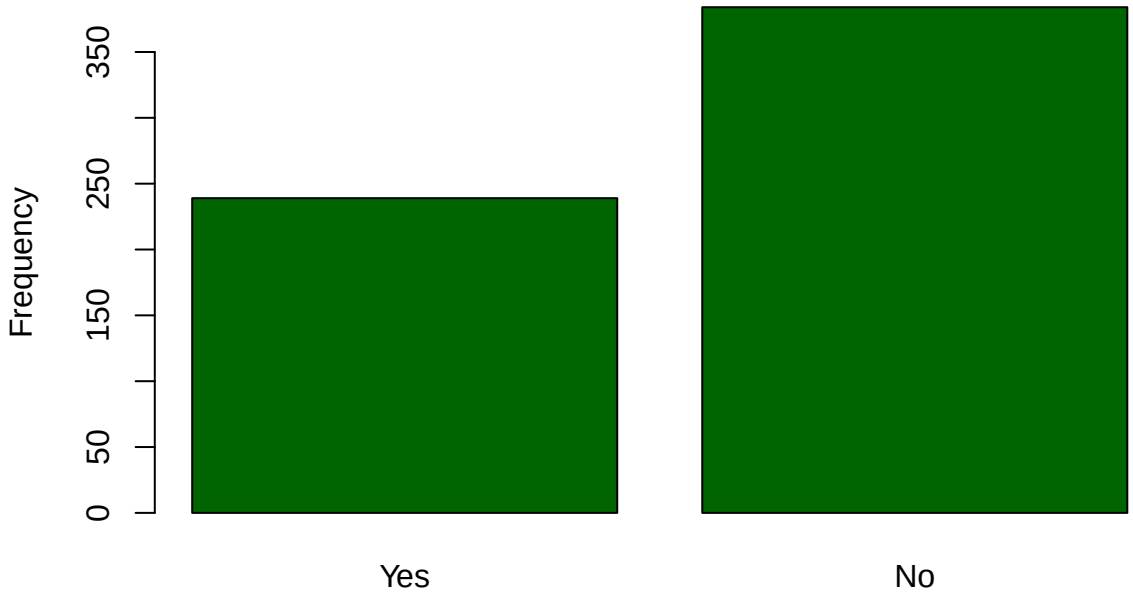
```
##   analysis assessment          n          p
##        623        268        891         12
```

According to the output dimensions of the split, there are 623 observations in the training set and 268 observations in the testing set.

Without using stratified sampling for the data, we might end up with a lot of observations in the training set where the passenger survived and very few where they did not, or vice versa. This means we would be building our model off of much more data for one outcome than the other, and would result in the model having low prediction accuracy and performing very poorly on the testing data set.

Question 2: Using the training data set, explain and describe the distribution of the outcome variable *survived*.

```
plot(titanic_train$survived, col= "darkgreen", ylab="Frequency")
```



According to the above histogram, the distribution of the outcome variable in the training set is that the majority of the passengers did not survive. We can see the exact distribution of observations in the table below.

```
table(titanic_train$survived)
```

```
##
## Yes  No
## 239 384
```

Since 239 of the 623 observations in the set survived, we can conclude that about 38.4% of the observations in the training data set have the *survived* outcome value "Yes", while the other 61.6% have the outcome value "No".

Question 3: Using the training data set, create a correlation matrix of all continuous variables. Create a visualization of the matrix, and describe any patterns you see. Are any predictors correlated with each other? Which ones, and in which direction?

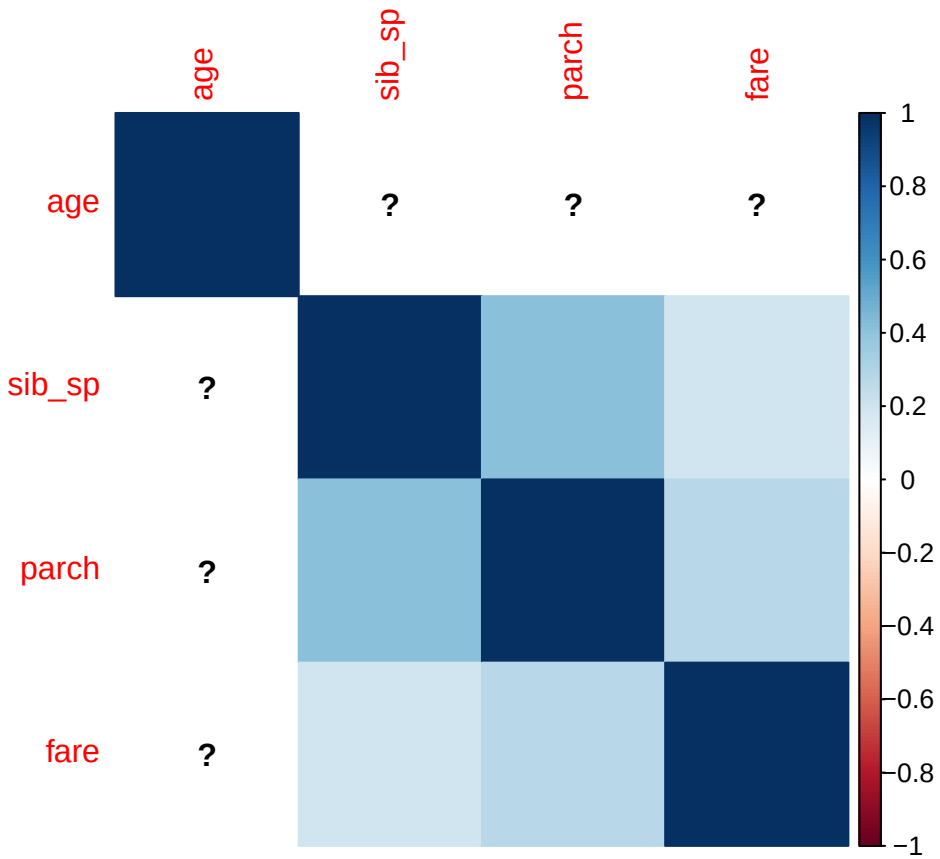
```
#first we will make a subset of the training set including only the continuous variables
titanic_train_cont <- titanic_train[c(6,7,8,10)]

#next we will make a correlation matrix for this newly created dataframe
cor(titanic_train_cont)
```

```
##         age    sib_sp     parch      fare
## age       1        NA        NA        NA
```

```
## sib_sp   NA 1.0000000 0.4136009 0.1958504
## parch    NA 0.4136009 1.0000000 0.2754689
## fare     NA 0.1958504 0.2754689 1.0000000
```

```
titanic_train_cont %>%
  cor() %>%
  corrplot(method='color')
```



Age does not seem to be significantly correlated to any of the other continuous predictor variables, but number of siblings/spouses on board (sip_sp) and number of parents/children (parch) are significantly positively correlated. This is to be expected, as having a family member on board does indicate that a passenger might be traveling with their family.There is also a slight positive correlation between the passenger fare and the number of parents/children on board, and an even smaller correlation between fare and sib_sp.

Question 4: Use the training data to create a recipe predicting the outcome variable *survived*. Include the following predictors: ticket class, sex, age, number of siblings/spouses on board, number of parents/children on board, and passenger fare.

Recall that there are missing values for *age*. Add an imputation step using "step_impute_linear()" to deal with this. Then use "step_dummy" to dummy encode categorical predictors. Finally, include interactions between sex and passenger fare, and age and passenger fare.

```
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data=titanic_train) %>%
  step_impute_linear() %>%
  step_dummy(sex) %>%
  step_interact(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare)
titanic_recipe
```

```
## Recipe
```

```
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for <none>
## Dummy variables from sex
## Interactions with starts_with("sex"):fare
## Interactions with age:fare
```

Question 5: Specify a logistic regression model for classification using the "glm" engine, then create a workflow. Add your model and the appropriate recipe. Finally, use fit() to apply your workflow to the training data.

```
#Part 1: specify and store a logistic regression model using the glm engine
logr_model<-
  logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

#Part 2: create a workflow and add our model and recipe
titanic_flow <-
  workflow() %>%
  add_model(logr_model) %>%
  add_recipe(titanic_recipe)

#Part 3: create and store a fit() object that applies our workflow to the training data
titanic_fit <- fit(titanic_flow, titanic_train)
titanic_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##     (Intercept)          pclass2          pclass3              age
##      -4.4887437        1.2457558        2.4861996        0.0660727
##          sib_sp            parch             fare         sex_male
##       0.4176469        0.1668191       -0.0113452        2.2317920
```

```
## sex_male_x_fare       age_x_fare
##       0.0158832       -0.0001809
##
## Degrees of Freedom: 491 Total (i.e. Null);  482 Residual
##   (131 observations deleted due to missingness)
## Null Deviance:       666.2
## Residual Deviance: 431.1     AIC: 451.1
```

Question 6: Repeat question 5, but specify a linear discriminant analysis model for classification using the "MASS" engine.

```
discr_model <-
  discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

titanic_flow_ld <-
  workflow() %>%
  add_model(discr_model) %>%
  add_recipe(titanic_recipe)

titanic_fit_ld<- fit(titanic_flow_ld, titanic_train)
```

Question 7: Repeat question 5, but specify a quadratic discriminant analysis model for classification using the "MASS" engine.

```
quadr_discr_mod <-
  discrim_quad() %>%
  set_engine("MASS") %>%
  set_mode("classification")

titanic_flow_qd <-
  workflow() %>%
  add_model(quadr_discr_mod) %>%
  add_recipe(titanic_recipe)

titanic_fit_qd <- fit(titanic_flow_qd,titanic_train)
```

Question 8: Repeat question 5, but specify a naive Bayes model for classification using the "klaR" engine. Set the *usekernel* argument to FALSE.

```
naive_bayes_mod <- naive_Bayes() %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE) %>%
  set_mode("classification")


titanic_flow_nb <-
  workflow() %>%
  add_model(naive_bayes_mod) %>%
  add_recipe(titanic_recipe) %>%
  add_step(drop_na(data=titanic_train))


titanic_fit_nb <- fit(titanic_flow_qd,titanic_train)
```

Question 9: Now you've fit four different models to your training data. Use predict() and

bind_cols() to generate predictions using each of these 4 models and your training data. Then use the *accuracy* metric to assess the performance of each of the four models. Which model achieved the highest accuracy on the training data?

```
#first we will create a vector of predictions for our logistic reg. model
predict_lr <- predict(titanic_fit, new_data=titanic_train, type="class")

#next we will create the same predict() object for our linear discriminant analysis model
predict_lda <- suppressWarnings(predict(titanic_fit_ld, new_data=titanic_train, type="class"))

#now we will create predictions again for our quadratic discriminant analysis model
predict_qd <- suppressWarnings(predict(titanic_fit_qd, new_data=titanic_train, type="class"))

#finally we will create the predictions for our naive bayes model
predict_nb <- suppressWarnings(predict(titanic_fit_nb, new_data=titanic_train, type="class"))

#using bind_cols(), we create a tibble of the predictions of each model and the observed outcomes
binded_predict <- bind_cols(prediction1=predict_lr[1], prediction2=predict_lda[1], prediction3=predict_
```

```
## New names:
## * .pred_class -> .pred_class...1
## * .pred_class -> .pred_class...2
## * .pred_class -> .pred_class...3
## * .pred_class -> .pred_class...4
```

```
binded_predict
```

```
## # A tibble: 623 x 5
##    .pred_class...1 .pred_class...2 .pred_class...3 .pred_class...4 outcome
##    <fct>           <fct>           <fct>           <fct>           <fct>
##  1 No              No              No              No              No
##  2 <NA>            <NA>            <NA>            <NA>            No
##  3 No              No              No              No              No
##  4 No              No              No              No              No
##  5 No              No              No              No              No
##  6 No              No              No              No              No
##  7 Yes             Yes             No              No              No
##  8 No              No              No              No              No
##  9 <NA>            <NA>            <NA>            <NA>            No
## 10 No              No              No              No              No
## # ... with 613 more rows
```

Above is a tibble with 5 columns: one for the predicted outcomes of each of our 4 models, and one final column for the actual observed outcomes. Next we will compute the accuracy of each model and compare.

```
#compute the accuracy of the logistic regression model
lr_acc <- augment(titanic_fit,new_data=titanic_train) %>%
  accuracy(truth= survived, estimate= .pred_class)
print(lr_acc)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.823
```

```
#compute the accuracy of the linear discriminant analysis model
lda_acc <- suppressWarnings(augment(titanic_fit_ld,new_data=titanic_train) %>%
```

```
  accuracy(truth= survived, estimate= .pred_class))
print(lda_acc)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.801
```

```
#compute the accuracy of the quadratic discriminant analysis model
qd_acc <- suppressWarnings(augment(titanic_fit_qd,new_data=titanic_train) %>%
  accuracy(truth= survived, estimate= .pred_class))
print(qd_acc)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.780
```

```
#compute the accuracy of the naive bayes model
nb_acc <- suppressWarnings(augment(titanic_fit_nb,new_data=titanic_train) %>%
  accuracy(truth= survived, estimate= .pred_class))
print(nb_acc)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.780
```

From the accuracy estimates above, we can see that the logistic regression model has the highest accuracy.

Question 10: Fit the model with the highest training accuracy to the testing data. Report the accuracy of the model on the testing data. Again using the testing data, create a confusion matrix and visualize it. Plot an ROC curve and calculate the area under it (AUC). How did the model perform? Compare its training and testing accuracies. If the values differ, why do you think this is so?

```
#first we will use predict() to fit the model to the test data
new <- predict(titanic_fit, new_data= titanic_test,type="class")
```

```
#next we calculate the accuracy of the model on the testing data
augment(titanic_fit, new_data= titanic_test) %>%
  accuracy(truth= survived, estimate= .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.802
```

According to the code above, the accuracy of the logistic regression model on the testing data is about 0.802, while the accuracy of the model on the training data was about 0.823.The testing accuracy is, as expected, lower than the training accuracy, but still higher than the training accuracy of the QDA model. Now, we will compute a confusion matrix of the fitted data.

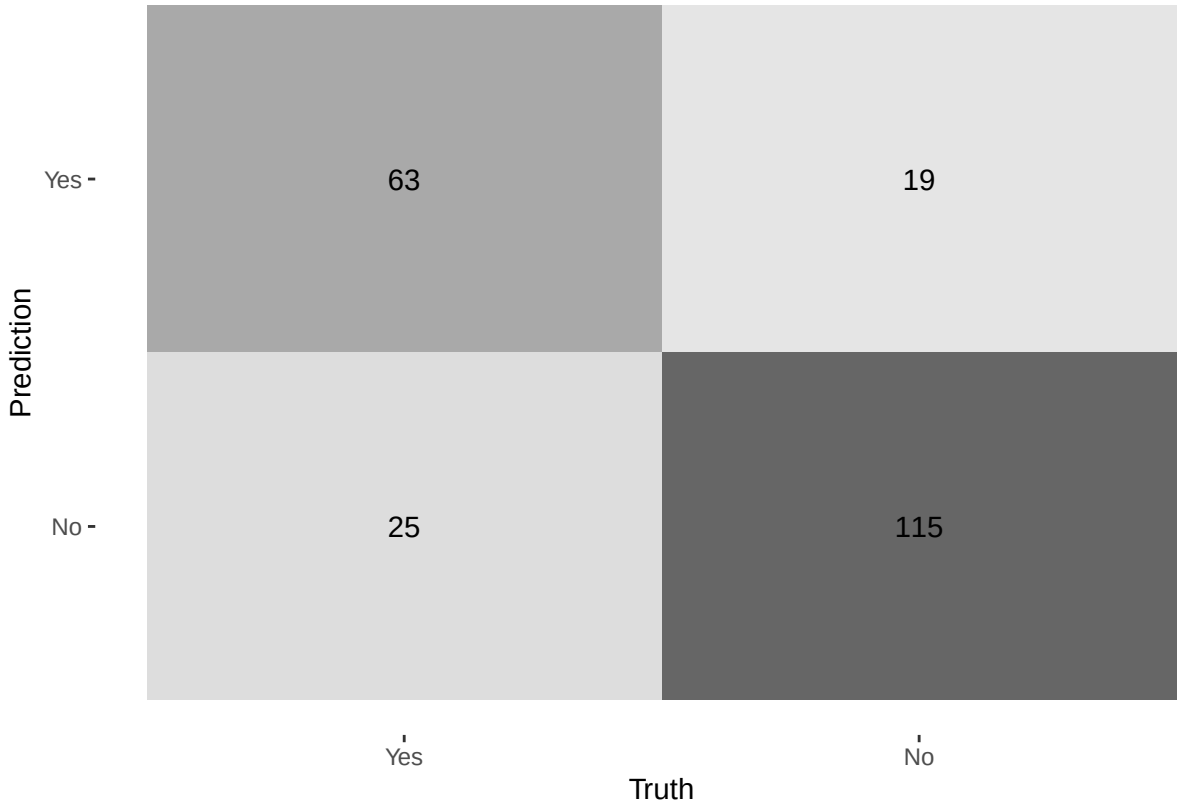
```
augment(titanic_fit, new_data=titanic_test) %>%
  conf_mat(truth= survived, estimate= .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes  63  19
```

```
##        No   25 115
```

Next, we will output a visual representation of the above matrix.

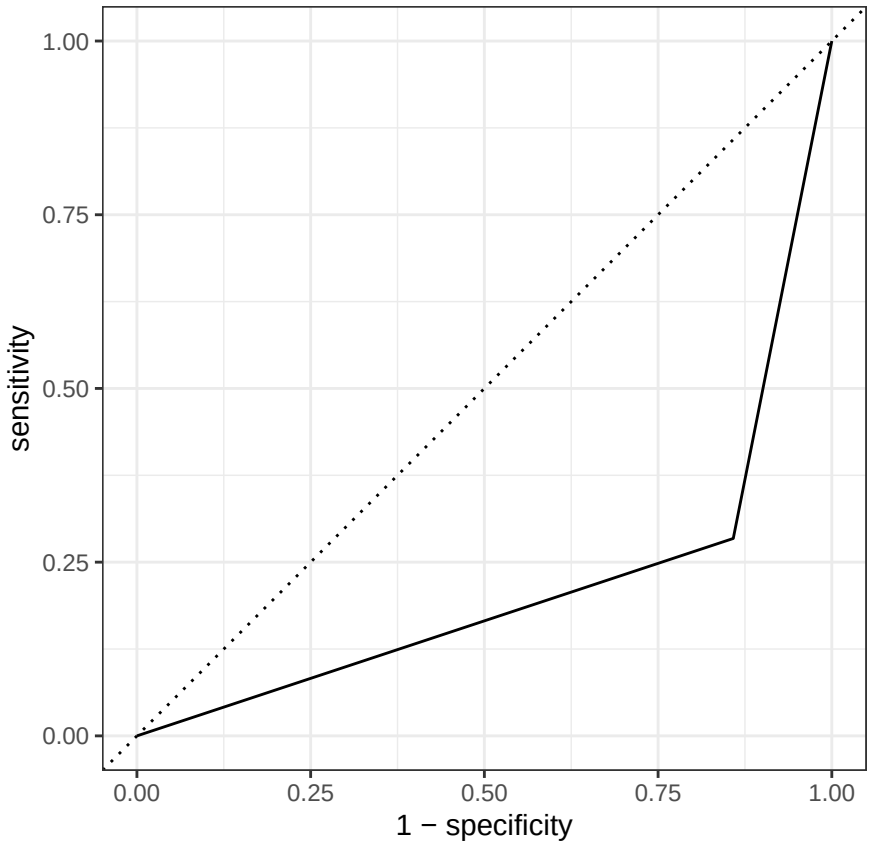
```
augment(titanic_fit, new_data=titanic_test) %>%
  conf_mat(truth= survived, estimate= .pred_class) %>%
  autoplot(type="heatmap")
```



From this image, we can see that the model is performing with pretty good accuracy; there are many more accurate predictions (the top left and bottom right squares) than inaccurate predictions. To be exact, almost three times as many "YES" observations were accurately predicted than not, and over six times as many "NO" observations were predicted correctly than incorrectly. Our next step in assessing the model is to plot an ROC curve.

```
final_fit <- augment(titanic_fit, new_data= titanic_test)
final_fit$.pred_class <- as.numeric(final_fit$.pred_class)

roc_curve(truth= factor(survived), estimate= .pred_class, data=final_fit) %>%
  autoplot()
```




```
roc_auc(data=final_fit, truth=factor(survived),estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.213
```

Since the curve is below the diagonal line, the model doesn't appear to distinguish between classes as well at different thresholds. A low AUC means a low measure of separability.

The testing accuracy we computed was lower than the training, since the model was fitted specifically to the training data and the test set is also noticably smaller, but the fitted model didn't appear to lose too much accuracy.